**1、本章面试题**

           springmvc如何上传文件或者多文件上传

springmvc如何统一处理异常

**2、知识点**

**2.1、课程回顾**

       拦截器（interceptor）的使用和作用

**2.2、本章重点**

         文件上传，统一异常处理

**3、具体内容**

**3.1springmvc 文件上传**

**3.1.1文件上传原理**

 使用最底层IO读写，把本地文件通过不同的上传方式上传到服务器或者是专门的文件服务器上（ftp服务器）

**3.1.2 springmvc实现文件上传过程**

需求：添加新闻时，为新闻添加新闻图片（每条新闻图片数量不固定）

-- 为新闻表添加主键

alter table tb\_news add constraint pk\_news\_id primary key(id);

-- 新建新闻附件表

create table tb\_news\_files(

   id number  primary key,

   name varchar2(100),

   path varchar2(500),

   newsid NUMBER(10),

   constraint fk\_newsid\_news\_id  foreign key(newsid) references tb\_news(id)

);

);

-- 创建附件序列

create sequence seq\_news\_files\_id;

**1,编写JSP页面**



**2,编写controller**

if(pic!=null){

//1，在项目根目录下创建uploadfiles目录，把相对路径转换绝对路径，为后面文件读写做准备

//request.getRealPath("uploadfiles");

String newFilePath = "uploadfiles/a/b/";

// servlet中全局域的对象 ServletContext 等于jsp中application

String realPath = request.getServletContext().getRealPath(newFilePath);

System.out.println("真实路径:"+realPath);

//获取文件的原来名称 originalFilename = a.jpg

String originalFilename = pic.getOriginalFilename();

//组装新的图片名称，避免不同用户上传相同名称图片时，产生覆盖

//获取源文件名称后缀 suffix = .jpg

String suffix = originalFilename.substring(originalFilename.lastIndexOf("."));

// 组装新名称

String newFileName = UUID.randomUUID()+suffix;

//按时间戳+随机数组装

/\* Random r = new Random();

Math.floor(Math.random()\*1000000)

String newFileNames = new SimpleDateFormat("yyyyMMddHHmm").format(new Date())

+r.nextInt(1000000)+suffix;\*/

//C:\Users\Administrator\.IntelliJIdea2017.2

// \system\tomcat\Unnamed\_ssm\_upload\_demo20191108

// /uploadfiles/a/b/c/c653c176-3ad0-4f44-8127-ef378989b587.jpg

File file = new File(realPath+"/"+newFileName);

//判断文件是否存在，如果不存在，创建父目录

if(!file.exists()){

//创建父目录

file.mkdirs();

}

try {

//文件读写 springmvc底层提供文件读写，把本地的pic读写到服务器的file对象

pic.transferTo(file);

} catch (IOException e) {

e.printStackTrace();

}

NewsFile newsFile =new NewsFile();

//设置上传文件的原始名称

newsFile.setName(originalFilename);

//设置上传后的服务器路径 //uploadfiles/a/b/c653c176-3ad0-4f44-8127-ef378989b587.jpg

newsFile.setPath(newFilePath+newFileName);

//设置上面添加新闻后返回的ID

newsFile.setNewsId(news.getId());

//保存附件

newsFileService.add(newsFile);

}

**3,上传配置**

<!--上传配置-->

<bean id="multipartResolver" class="org.springframework.web.multipart.commons.CommonsMultipartResolver">

<!--上传文件最大限制 单位是字节 大约10M 默认值2M-->

<property name="maxUploadSize" value="10240000"></property>

</bean>

<!-- 设定默认编码 -->

<property name="defaultEncoding" value="UTF-8"></property>

<!-- 设定文件上传的最大值为5MB，5\*1024\*1024 -->

<property name="maxUploadSize" value="5242880"></property>

<!-- 设定文件上传时写入内存的最大值，如果小于这个参数不会生成临时文件，默认为10240 -->

<property name="maxInMemorySize" value="40960"></property>

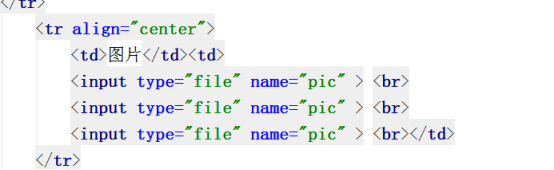
<!-- 上传文件的临时路径 -->

<property name="uploadTempDir" value="fileUpload/temp"></property>

<!-- 延迟文件解析 -->

<property name="resolveLazily" value="true"/>

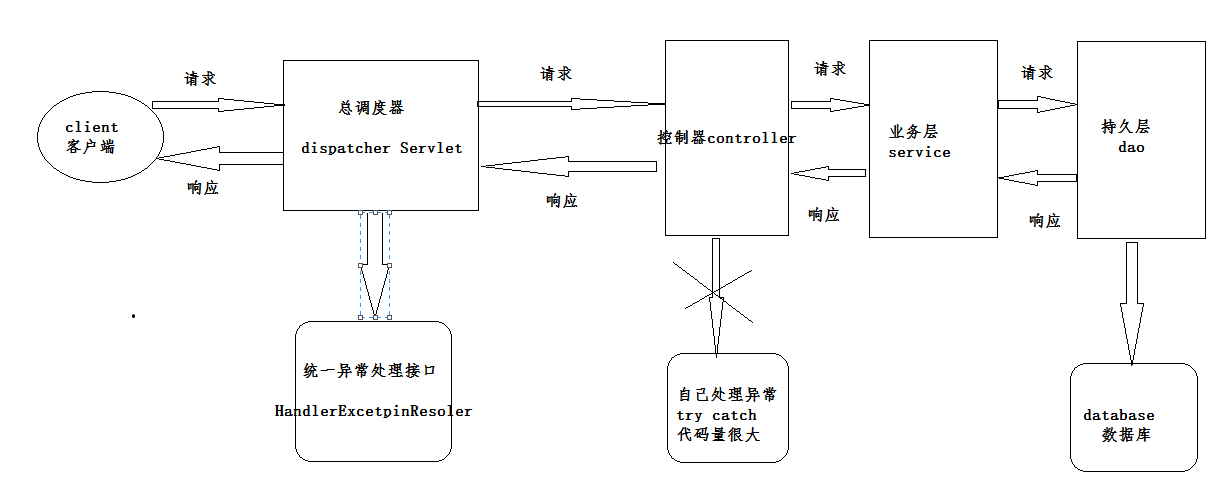
**多文件上传：**





**3.2 springmvn异常处理**

**3.2.1 统一异常流程图**



**3.2.2 实现方式**

1,配置文件方式

<bean class="org.springframework.web.servlet.handler.SimpleMappingExceptionResolver">

<!--所有异常默认的异常处理页面-->

<property name="defaultErrorView" value="error"></property>

<!--异常页面获取异常信息的名称-->

<property name="exceptionAttribute" value="e"></property>

</bean>

2,编写异常类的方式

public class GlobalException implements HandlerExceptionResolver {

@Override

public ModelAndView resolveException(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse, Object o, Exception e) {

ModelAndView modelAndView = new ModelAndView();

modelAndView.setViewName("error");

// System.out.println(o.getClass().getName()+"...........");

modelAndView.addObject("e",e);

return modelAndView;

}

}

自定义异常：

public class CustomException extends Exception {

public CustomException(){};

public CustomException(String message){

super(message);

};

}

错误页面:

对不起，出错了，请联系管理员，管理员联系方式：李工，电话：13812341234。。。。

<%

Exception ex = (Exception)request.getAttribute("e");

out.print("错误描述：<br>"+ex.getMessage()+"<br>");

ex.printStackTrace(new java.io.PrintWriter(out));

%>

**4、本章总结**

**4.1 总结本章知识点**

**springmvc流程原理和MultipartResolver接口：**

在DispatcherServlet中定义了一个MultipartResolver属性，如果用户配置了该Bean，启动容器的时候，会自动注入参数，如果用户没有配置，则默认为null。当DispaterServlet收到请求时，它的checkMultipart()方法会调用MultipartResolver的isMultipart()方法判断请求中是否包含了文件且multipartResolver属性存在实例。如果满足条件，则调用MultipartResolver的resolveMultipart()方法对请求数据进行解析，然后将文件数据解析成MultipartFile并封装在MultipartHttpServletRequest对象中返回给DispatcherServlet。解析之后的MultipartFile 封装了请求数据中的文件，此时这个文件存储在内存中或临时的磁盘文件中，需要将其转存到一个合适的位置，因为请求结束后临时存储将被清空。

**4.2 面试题答案**

**4.3 预习下一章重点**